Escuela Politécnica Superior

Creación de una red social para enfermos y familiares con enfermedades raras

Grado en Ingeniería Informática

Trabajo Fin de Grado

Autor:
Álvaro Martín Resel

Tutor/es:
Miguel Ángel Cazorla Quevedo

Junio 2020

Universitat d'Alacant
Universidad de Alicante

Creación de una red social para enfermos y familiares con enfermedades raras

Autor

Álvaro Martín Resel

Tutor/es

Miguel Ángel Cazorla Quevedo

Ciencias de la Computación e Inteligencia Artificial (CCIA)

GRADO EN INGENIERÍA INFORMÁTICA

Escuela Politécnica Superior

Universitat d'Alacant
Universidad de Alicante

ALICANTE, Junio 2020

Preámbulo

En este proyecto se va a realizar una aplicación web que tiene como finalidad ayudar a las personas que sufren de enfermedades raras. Dado que no hay ninguna plataforma en el mercado que aglutine las características que se proporcionan en este proyecto, proporcionando a los pacientes información sobre sus patologías y dándoles la oportunidad para que se pongan en contacto con otras personas de todas partes del mundo que sufren las mismas enfermedades que ellos.

Agradecimientos

En este apartado, me gustaría agradecer a todas las personas que me han acompañado durante esta etapa de mi vida. Estos años en la universidad han sido años con mezcla de emociones. Por una parte, años duros, de mucho sacrificio debido al grado. Pero, por otra parte, han sido años donde he aprendido mucho, de los que me llevo infinidad de experiencias positivas y que, a su vez, me han ayudado a crecer para mi futuro profesional y, lo más importante, como persona.

Este trabajo de fin de grado es la culminación a todo esto, y por ello me gustaría agradecer a mis compañeros de carrera, convertidos en grandes amigos, que tanto me han ayudado, así como a mi familia y a mis amigos de toda la vida por apoyarme cuando puede que, en ocasiones, no haya sido lo más fácil. Sin olvidarme por supuesto, de mi tutor de este trabajo de fin de grado, que tanto me ha ayudado para poder sacar esto adelante, así como al resto de docentes que me han ayudado durante este camino.

A todos vosotros, muchas gracias.

Índice general

Índice de figuras

1 Introducción

Las enfermedades raras (ER) o poco frecuentes son aquellas que tienen una baja prevalencia en la población. Cada país tiene unas medidas diferentes para considerar rara a una enfermedad. En Europa, para ser considerada como rara, cada enfermedad específica sólo puede afectar a un número limitado de personas. Concretamente, cuando afecta a 1 de cada 2.000 habitantes.

Sin embargo, las patologías poco frecuentes afectan a un gran número de personas, ya que según la Organización Mundial de la Salud (OMS), existen cerca de 7.000 enfermedades raras que afectan al 7% de la población mundial. En total, se estima que en España existen más de 3 millones de personas con enfermedades poco frecuentes.[1]

Una enfermedad puede ser rara en una región, pero habitual en otra. Este es el caso de la talasemia, una anemia de origen genético, que es rara en el Norte de Europa, pero frecuente en la región del Mediterráneo. La "enfermedad periódica" es rara en Francia, pero común en Armenia. También existen muchas enfermedades comunes cuyas variantes son raras.[2]

Incluso en la actualidad, con la gran cantidad de medios sanitarios que se poseen y las tantas investigaciones que se han hecho, las personas que sufren enfermedades raras siguen teniendo muchos problemas. Estos problemas surgen sobre todo del desconocimiento que hay sobre estas patologías. Patologías que debido al ínfimo porcentaje de personas que las sufren no son investigadas ni prácticamente estudiadas por los profesionales sanitarios en general.

[1] FEDER: www.enfermedades-raras.org
[2] Orphanet: www.orpha.net

Todo este cóctel de circunstancias hace que las personas que las sufren vivan un auténtico calvario esperando años hasta ser diagnosticados, teniendo que desplazarse a otros lugares del país o del mundo para ello o buscando un tratamiento eficaz que les ayude a combatir la patología que padecen.

Muchas veces incluso diagnosticados y con tratamiento aplicable sufren de la falta de información ya que no a todas las personas les afectan igual estas patologías, y al no haberse estudiado tanto por la falta de casos, hace que en muchas ocasiones las personas que las padecen se encuentren perdidos sin saber qué hacer o a quién recurrir.

1.1 Justificación

Dadas las circunstancias de las personas que sufren alguna de las muchas enfermedades raras existentes, en muchas ocasiones carecen de información suficiente sobre cómo tratarse o incluso no encuentran especialistas que puedan aportar un poco de luz a su situación.

A pesar de que hay algunas plataformas existentes que intentan resolver esta necesidad, y podrían ayudar de cierta manera, nos encontramos con que ninguna de ellas la resuelve con éxito, ya que todas ellas presentan diferentes carencias: algunas de ellas son proyectos pilotos con pocas comunidades, otras son meros lugares de opinión, también denominados foros, donde las personas hacen preguntas y reciben respuestas concretas de otros usuarios.

En general, nos encontramos plataformas con un número muy reducido de usuarios, poca experiencia de usuario y, lo más importante, no resuelven las necesidades de las personas.

1.2 Objetivos

El objetivo principal de este proyecto es crear una plataforma que ayude a las personas con enfermedades raras a obtener información, tanto oficial como de otros pacientes con sus mismas patologías sobre éstas. La plataforma se llamará Finder.

Para llevar a cabo este objetivo se van a llevar a cabo las siguientes tareas o sub-objetivos:

- Hacer un estudio de mercado sobre plataformas similares para analizar la viabilidad de la plataforma.
- Estudiar las tecnologías elegidas para crear la aplicación web.
- Diseñar un esquema Entidad-Relación del que se creará la aplicación.
- Analizar de dónde sería posible obtener los datos necesarios.
- Crear el API al que se harán consultas desde el frontend.
- Crear el frontend con un diseño intuitivo.

2 Estado del arte

En este apartado vamos a analizar el mercado actual que intenta cumplir las necesidades expuestas previamente. Hay una serie de plataformas que vamos a desglosar una por una, analizando tanto sus puntos fuertes como sus puntos débiles para este estudio. Posteriormente, sacaremos unas conclusiones de todas ellas en conjunto.

Share4rare

Es un proyecto impulsado por el Hospital Sant Joan de Déu, en Esplugues de Llobregat (Barcelona) financiado con fondos de la unión europea. Este proyecto es aún un proyecto piloto con un número muy reducido de comunidades, a las que por el momento no es posible añadir más.

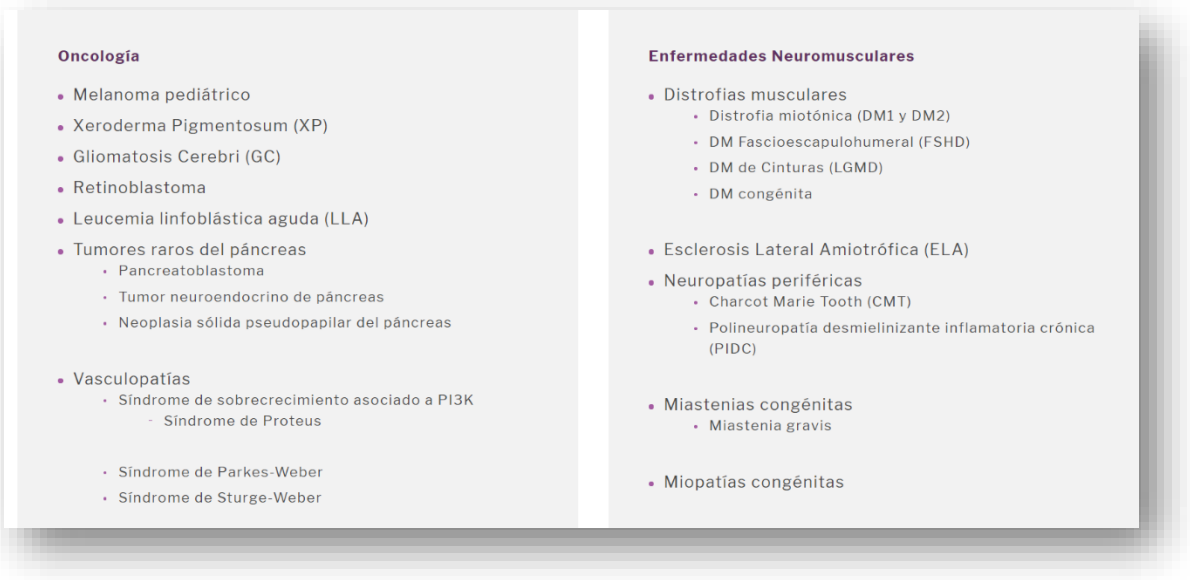

Figura 2.1: Captura de pantalla Share4rare

A pesar de tener un foro, poco intuitivo ya que las preguntas no están separadas por comunidades, donde los usuarios pueden comunicarse entre sí, está diseñada para estudios profesionales en vez de opiniones.

RareConnect

En esta plataforma se cumplen algunos de los requisitos que queremos implementar para Finder. Sin embargo, hay otros que consideramos importantes que no. RareConnect es básicamente un sistema de comunidades, cada comunidad es para una enfermedad y contiene un foro para la discusión de los usuarios, pero carece de información general sobre cada patología, además de que hay algunas otras funcionalidades mejorables como la comunicación directa mediante usuarios que se hace vía email o la búsqueda que no funciona.

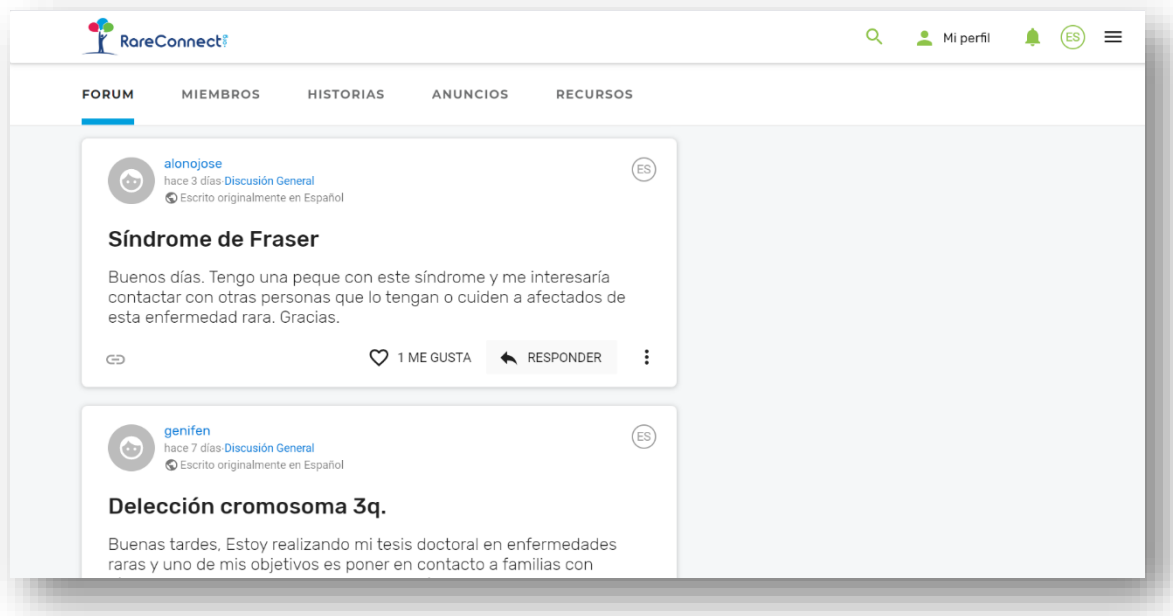

Figura 2.2: Captura de pantalla RareConnect

mydisease2ez

Esta plataforma, en general, parece peor desarrollada que las dos anteriores en temas de diseño y usabilidad de usuario. Está dividida por comunidades que contienen un foro y una

pequeña descripción. Sin embargo, para el estudio, he sido dado de alta en una comunidad y aparece con 0 entradas al foro y una descripción vacía. También es destacable comentar que la web, supuestamente en inglés y francés, cambia a francés cuando se hace click a cualquier enlace.

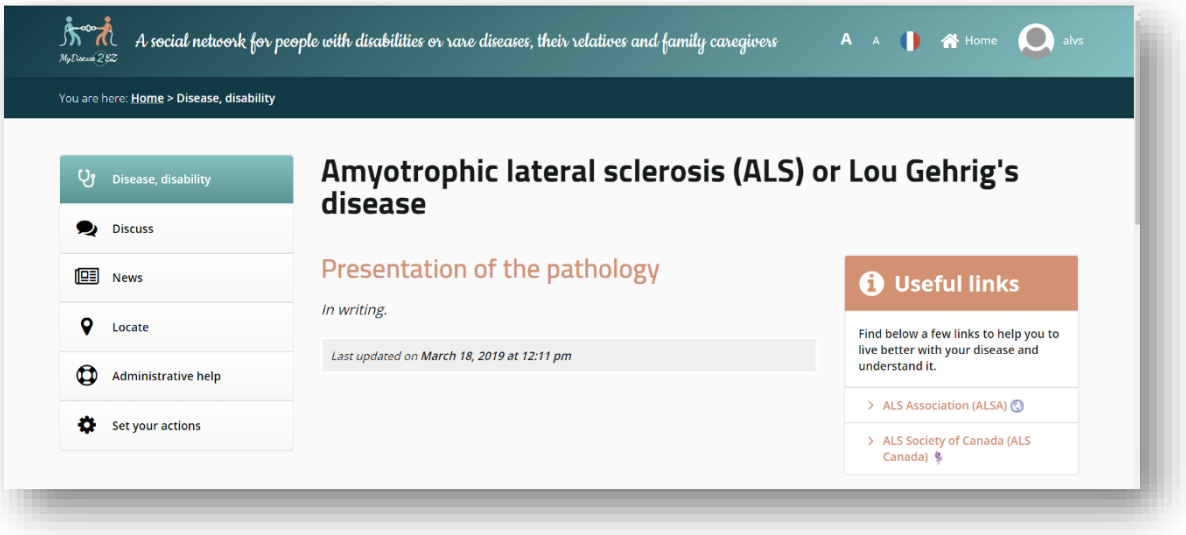

Figura 2.3: Captura de pantalla mydisease2ez

En conclusión, estas plataformas analizadas no cubren las necesidades que una persona con una enfermedad rara podría tener, tienen muy pocos usuarios en general, lo que hace su utilidad se reduzca drásticamente al estar pensadas para que las personas compartan información entre sí. Además, cuentan con una no muy buena experiencia de usuario.

3 Metodología

En este apartado se va a exponer tanto las tecnologías y herramientas como las metodologías que se han utilizado para la realización del proyecto.

3.1 Aproximación

A pesar de ser únicamente un desarrollador de software, para la realización de este proyecto se han utilizado metodologías ágiles. Principalmente sprints semanales para los que se han definido una serie de tareas a realizar al principio de la semana. Otra metodología ágil que se ha seguido es el tablero KANBAN, dónde se han ido introduciendo las tareas, para llevar un control sobre éstas, agrupadas en tres columnas (To do, In progress y Done).

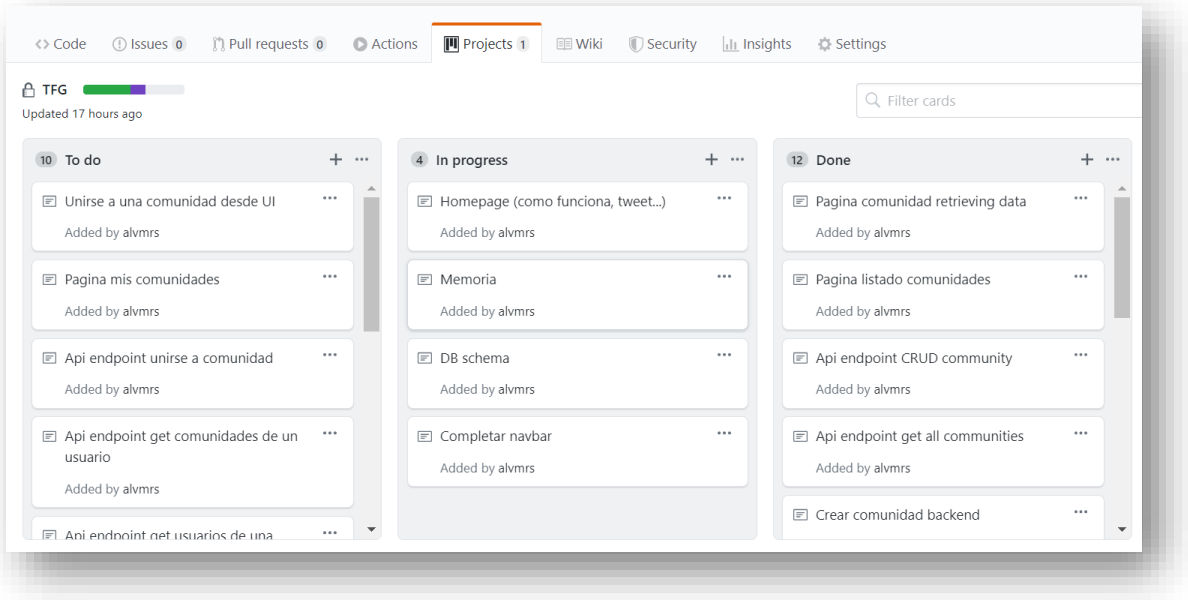

Figura 3.1: Tablero KANBAN de GitHub Project utilizado

Otras características identificadas como ágiles que se han seguido en este proyecto han sido el desarrollo evolutivo y flexible, así como adaptabilidad para adecuar el proyecto dependiendo de las circunstancias requeridas en cada momento del desarrollo.

3.2 Herramientas utilizadas

Laravel

Laravel es un framework para desarrollo web basado en PHP. Aporta una sintaxis que facilita la integración en el backend, especialmente en las consultas a la base de datos, así como para la creación de APIs. También dispone de parte de frontend basada en plantillas Blade, sin embargo, en este proyecto usaremos Vue.js en el lado del cliente.

Vue.js

Vue.js es un framework progresivo muy potente en el lado del cliente basado en JavaScript, similar a React.js. Nacido en 2014, es una tecnología que está en auge y cada vez se utiliza más para desarrollar aplicaciones. Al ser progresivo, se puede utilizar tanto para aplicaciones pequeñas como para otras más grandes sin cambiar el workflow, siempre con una buena experiencia de desarrollo y un gran rendimiento.

Dispone de una gran cantidad de librerías que hacen que pueda ser aún mucho más potente como Vuex para gestionar el estado de la aplicación, vue-router para gestionar las rutas o vue-cli para iniciar el proyecto de una nueva aplicación de una forma sencilla y sin tener que preocuparse de la configuración. Además, Vue.js se integra perfectamente con Laravel.

Vuetify

Vuetify es un framework basado en la estética de Material Design para aportar estilo a las aplicaciones creadas en Vue.js. Es muy completa y útil ya que dispone de gran cantidad de componentes que incluye desde algunos básicos como botones o formularios a otros más complejos como Carousels, Dialogs o Snackbars.

MySQL

MySQL es un sistema de gestión de bases de datos relacionales de código abierto basado, como su nombre indica en SQL. Inicialmente una compañía sueca, actualmente es

propiedad del gigante de las bases de datos Oracle desde que adquirió Sun Microsystems en 2010.

Git

Git es el sistema de control de versiones más popular y expandido. Es de código abierto y se utiliza para realizar un seguimiento de los cambios del código de un proyecto a lo largo de éste. Fue diseñado por Linus Torvalds en 2005, que buscaba un sistema distribuido de código abierto que fuera similar a BitKeeper.

GitHub

Github es una plataforma colaborativa para alojar proyectos software que utilizan el sistema de control de versiones Git. Es la más extendida y fue comprada por Microsoft en 2018. Además, GitHub ha ido incorporando otras características muy útiles como GitHub Project, que se utiliza para administrar las diferentes tareas por medio de un tablero KANBAN.

LucidChart

LucidChart es un espacio de trabajo colaborativo para crear todo tipo de diagramas, así como para visualizar datos. Se ha utilizado en este proyecto para crear el esquema Entidad-Relación de la base de datos.

4 Desarrollo

En este apartado se va a exponer todo el proceso de desarrollo que se ha seguido en este proyecto. Todos los elementos que han sido desarrollados, así como la forma en que se han integrado unos con otros.

4.1 Integración

Para la correcta integración de los componentes de este proyecto, se ha decidido optar por una arquitectura completamente desacoplada siguiendo el modelo MVC (Modelo, Vista, Controlador).

Para ello, como se ha mencionado en las tecnologías utilizadas, se ha creado un backend con el framework Laravel, incluyendo un API, que será el punto de acceso al backend desde el frontend, desarrollado en Vue.js.

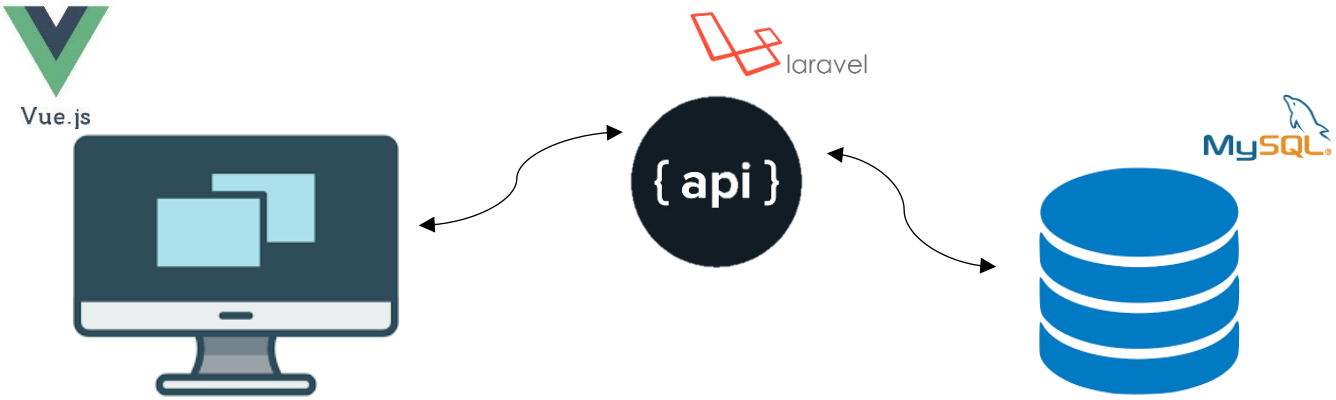

Figura 4.1: Esquema de arquitectura del proyecto

Esta arquitectura está totalmente desacoplada entre backend y frontend, ya que no tienen ningún tipo de dependencia, el backend está construido por una parte y el frontend por otra. Para comunicarse, simplemente el frontend hace llamadas al API, que hace las consultas en la base de datos y devuelve los resultados. Esto hace que la plataforma sea completamente escalable, ya que podrían crearse otras aplicaciones, como por ejemplo aplicaciones móviles para iOS y Android que hicieran llamadas al API de la misma forma que lo hace la aplicación web y no haría falta modificar nada del backend para su correcto funcionamiento.

También, para que el desarrollo de la plataforma sea posible, ha sido necesario diseñar como parte del backend, un esquema de base de datos representado en el siguiente esquema Entidad-Relación:

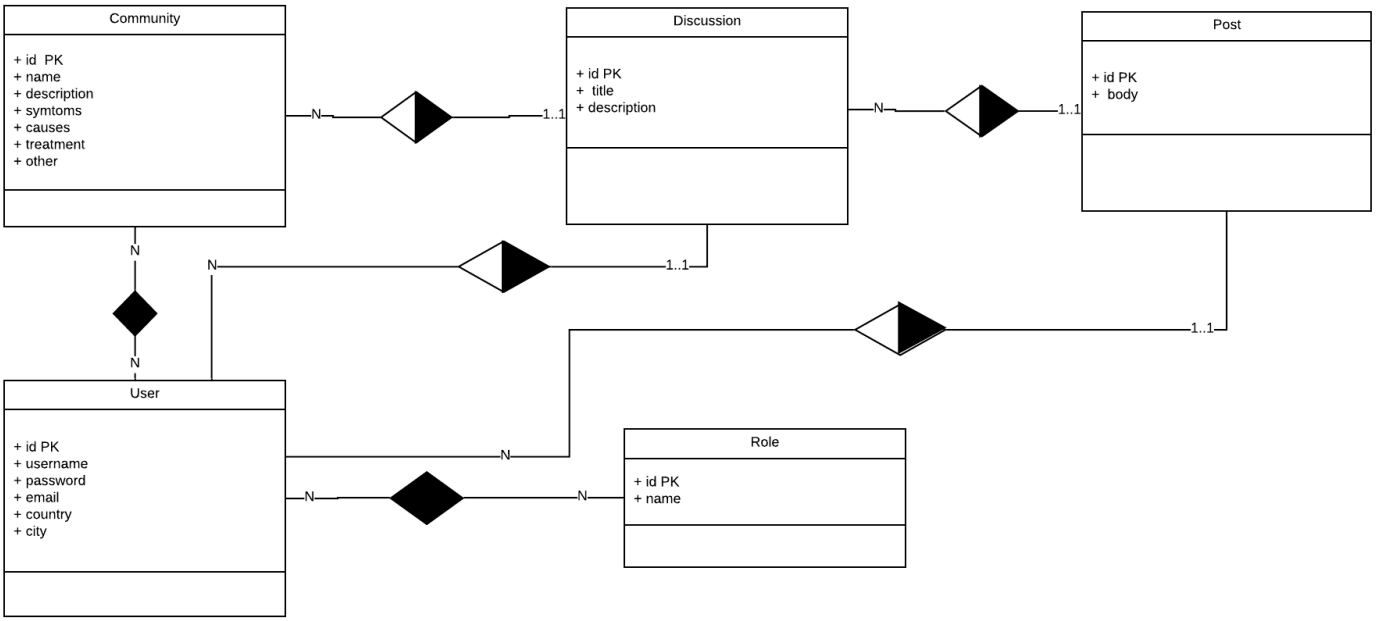

Figura 4.2: Diagrama Entidad-Relación de la base de datos

4.2 Funcionalidad

En este apartado se van a describir todos los elementos desarrollados en el proyecto, que principalmente, están divididos en dos partes: el API, desarrollado en Laravel y la interfaz de usuario desarrollada en Vue.js, utilizando también Vuetify, como se ha mencionado anteriormente.

Como toda aplicación web, está dividida en dos partes: parte pública y parte privada, es decir una parte accesible a cualquier usuario que acceda a la web y otra exclusiva para usuarios que se hayan dado de alta. La mayoría de información general será pública, en la que se incluiría la página principal, comunidades e información detallada de cada comunidad. Sin embargo, hay elementos que serán privados como los foros de las comunidades, así como la información propia de un usuario, como es obvio.

4.2.1 API

Rutas

Todas las rutas del api comenzarán con /api. Además, a las rutas de autenticación se le añadirá /auth. También es remarcable destacar que tanto los cuerpos de las peticiones que se requieran como las respuestas por parte del API serán en formato JSON.

POST /login: Ruta de autenticación. Se utiliza cuando se quiere iniciar sesión, se mandan las credenciales del usuario en el cuerpo de la llamada. Si las credenciales son correctas se genera un token del tipo JWT y se le envía al usuario en la respuesta, en caso de que no sean correctas se le manda de respuesta que no está autorizado.

POST /signup: Ruta de autenticación. Se utiliza cuando se quiere registrar un usuario, mandan los datos del nuevo usuario en el cuerpo de la llamada. Si los datos son correctos, se informa en la respuesta de que se ha registrado al usuario con éxito. En caso contrario, se informa de que ha habido un error.

GET /logout: Se utiliza para que un usuario cierre sesión. Se borra el token almacenado en el navegador del usuario.

GET /user: Esta ruta obtiene los datos del usuario que tiene la sesión iniciada actualmente.

GET /communities: Esta ruta obtiene un listado de todas las comunidades existentes en la base de datos.

GET /communities/{id}: Esta ruta obtiene los datos de una de las comunidades existentes. Se pasa por parámetro el id de la comunidad.

POST /communities: Esta ruta crea una nueva comunidad en la base de datos. Para ello habrá que incluir los datos de la nueva comunidad en el cuerpo de la llamada. En caso de que no haya ningún error, así lo notifica. En caso contrario, se informa de que ha habido un error.

PUT /communities/{id}: Esta ruta modifica una comunidad existente en la base de datos. Para ello habrá que incluir los datos de la comunidad actualizada en el cuerpo de la llamada. Se pasa por parámetro el id de la comunidad a actualizar. En caso de que no haya ningún error, así lo notifica. En caso contrario, se informa de que ha habido un error.

DESTROY /communities/{id}: Esta ruta elimina una comunidad de la base de datos. Se pasa por parámetro el id de la comunidad a borrar.

GET /community/{id}/users: Esta ruta obtiene una lista de todos los usuarios que se han unido a una comunidad. Se pasa por parámetro el id de la comunidad a consultar.

GET /community/{id}/discussions: Esta ruta obtiene una lista de todas las discusiones que se han creado en una comunidad. Se pasa por parámetro el id de la comunidad a consultar.

GET /users: Esta ruta obtiene un listado de todos los usuarios existentes en la base de datos.

GET /users /{id}: Esta ruta obtiene los datos de uno de los usuarios existentes. Se pasa por parámetro el id del usuario.

POST /users: Esta ruta crea un nuevo usuario en la base de datos. Para ello habrá que incluir los datos del nuevo usuario en el cuerpo de la llamada. En caso de que no haya ningún error, así lo notifica. En caso contrario, se informa de que ha habido un error.

PUT /users/{id}: Esta ruta modifica un usuario existente en la base de datos. Para ello habrá que incluir los datos del usuario actualizado en el cuerpo de la llamada. Se pasa por parámetro el id del usuario a actualizar. En caso de que no haya ningún error, así lo notifica. En caso contrario, se informa de que ha habido un error.

DESTROY /users/{id}: Esta ruta elimina un usuario de la base de datos. Se pasa por parámetro el id del usuario a borrar.

GET /user/{id}/communities: Esta ruta obtiene una lista de todas las comunidades a las que se ha unido un usuario. Se pasa por parámetro el id del usuario a consultar.

POST /user/{id_user}/community/{id_community}: Esta ruta se utiliza para unir un usuario a una comunidad. Para ello, será necesario que se pase por parámetro tanto el id del usuario como el id de la comunidad.

GET /user/{id}/role: Esta ruta obtiene el rol del usuario del que se pasa el id por parámetro.

GET /discussions: Esta ruta obtiene un listado de todas las discusiones existentes en la base de datos.

GET /discussions/{id}: Esta ruta obtiene los datos de una de las discusiones existentes. Se pasa por parámetro el id de la discusión.

POST /discussions: Esta ruta crea una nueva discusión en la base de datos. Para ello habrá que incluir los datos de la nueva discusión en el cuerpo de la llamada. En caso de que no haya ningún error, así lo notifica. En caso contrario, se informa de que ha habido un error.

PUT /discussions/{id}: Esta ruta modifica una discusión existente en la base de datos. Para ello habrá que incluir los datos de la discusión actualizada en el cuerpo de la llamada. Se pasa por parámetro el id de la discusión a actualizar. En caso de que no haya ningún error, así lo notifica. En caso contrario, se informa de que ha habido un error.

DESTROY /discussions/{id}: Esta ruta elimina una discusión de la base de datos. Se pasa por parámetro el id de la discusión a borrar.

GET discussion/{id}/posts: Esta ruta obtiene todos los posts que han sido creados dentro de una discusión. Se pasa por parámetro el id de la discusión a consultar.

GET /posts: Esta ruta obtiene un listado de todos los posts existentes en la base de datos.

GET /posts/{id}: Esta ruta obtiene los datos de uno de los posts existentes. Se pasa por parámetro el id del post.

POST /posts: Esta ruta crea un nuevo post en la base de datos. Para ello habrá que incluir los datos del nuevo post en el cuerpo de la llamada. En caso de que no haya ningún error, así lo notifica. En caso contrario, se informa de que ha habido un error.

PUT /posts/{id}: Esta ruta modifica un post existente en la base de datos. Para ello habrá que incluir los datos del post actualizado en el cuerpo de la llamada. Se pasa por parámetro el id del post a actualizar. En caso de que no haya ningún error, así lo notifica. En caso contrario, se informa de que ha habido un error.

DESTROY /posts/{id}: Esta ruta elimina un post de la base de datos. Se pasa por parámetro el id del post a borrar.

Seguridad

Respecto a la seguridad del API, se ha utilizado el sistema de Json Web Tokens (JWT) para su segurización. Cuando se produce la autenticación, es decir, cuando un usuario inicia sesión, se le otorga un token que debe mandar en cada llamada que haga al API para poder obtener la información para la que es necesario iniciar sesión para obtenerla, si no lo incluyera se le deniega el acceso.

Obviamente, cuando el usuario inicia sesión a través de la web no tiene que preocuparse de estos aspectos, ya que la aplicación web está hecha de tal forma que el token queda almacenado y es enviado automáticamente cuando se haga una llamada al API. Sin embargo,

es importante para prevenir que cualquier persona no autorizada pueda acceder a la información de la base de datos mediante el API.

4.2.2 User Interface

Rutas

/: Esta ruta muestra la homepage de la aplicación web. En esta página se expone la plataforma, se explica su funcionamiento y se intenta atraer a futuros nuevos usuarios a que se den de alta.

Figura 4.3: Capturas de la página principal

/login: Esta ruta muestra la página de login para que los usuarios inicien sesión en la plataforma.

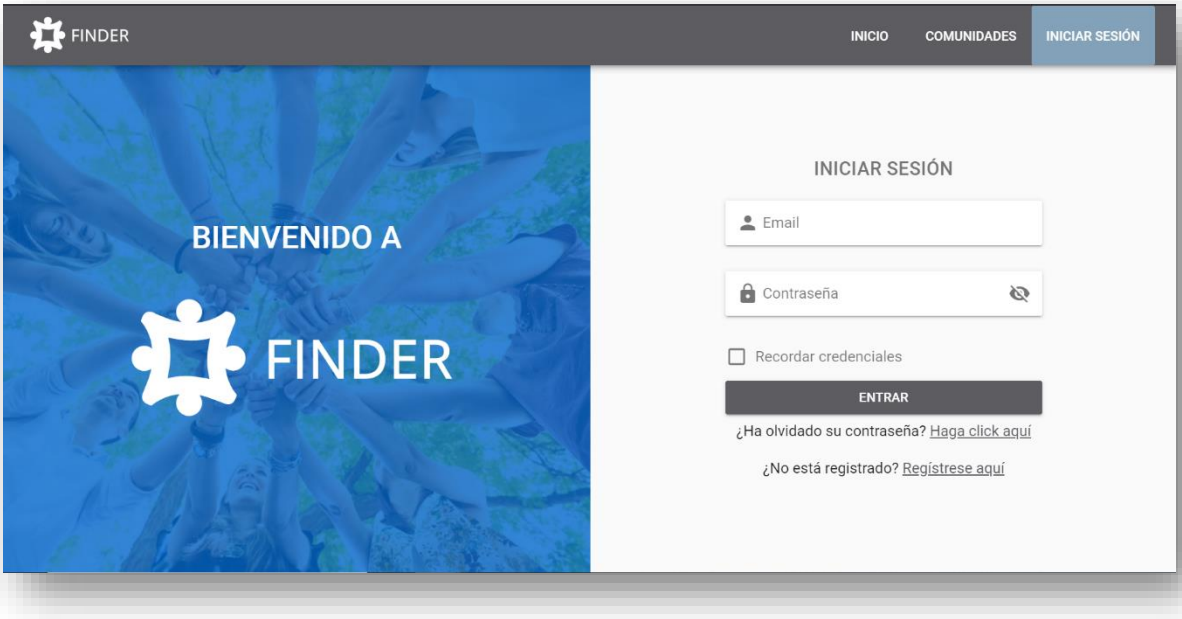

Figura 4.4: Página de inicio de sesión

/signup: Esta ruta muestra la página de signup para que los usuarios se den de alta en la plataforma.

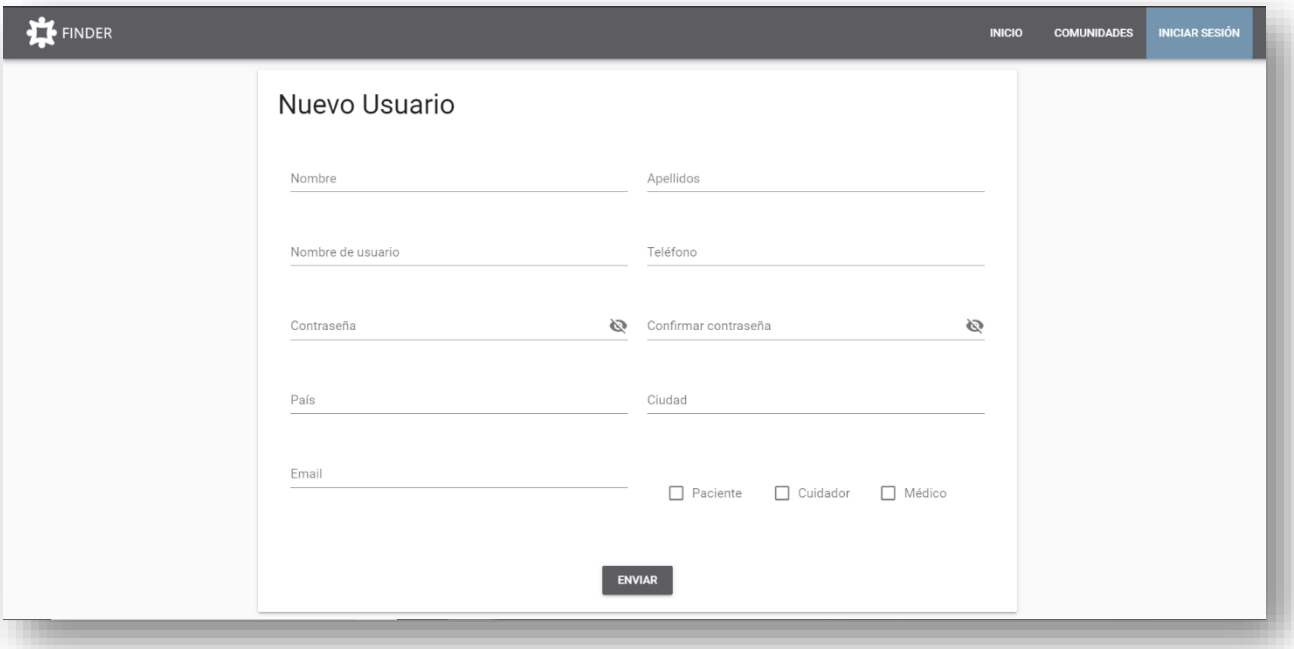

Figura 4.5: Página para crear un nuevo usuario

/profile: Esta ruta muestra una página en la que se incluya, como su nombre indica, un perfil del usuario que ha iniciado sesión, donde se muestran todos sus datos.

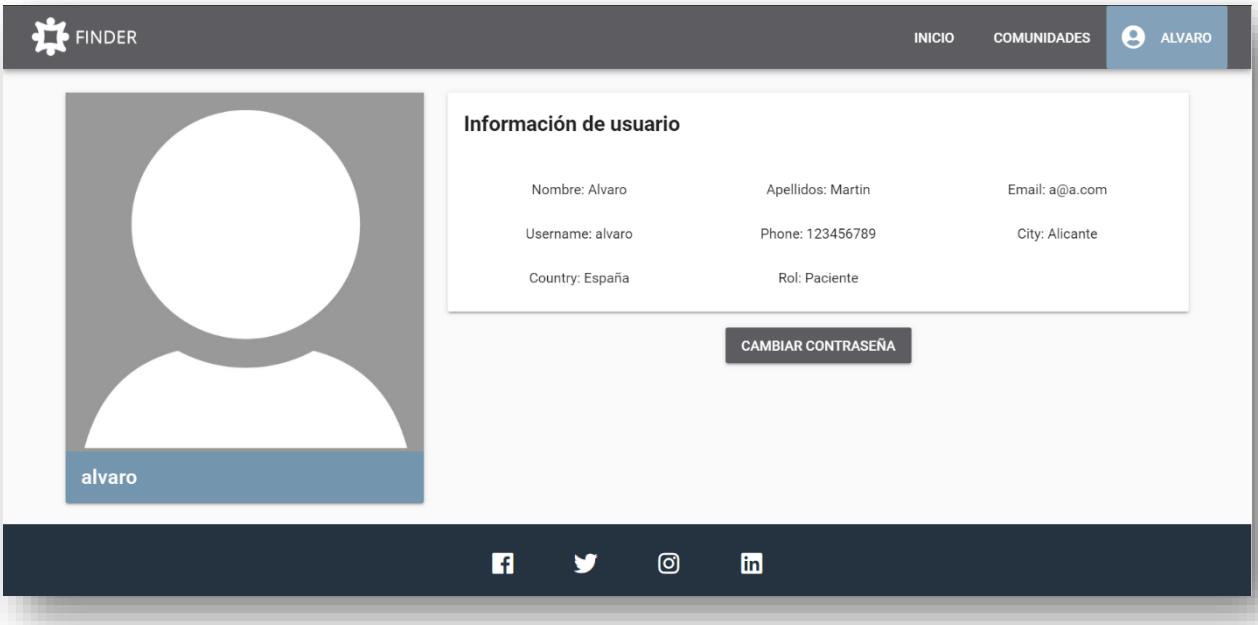

Figura 4.6: Página de perfil

/communities: Esta ruta muestra una página que muestra un listado de todas las comunidades actuales existentes en la aplicación. Hay un buscador que filtra las comunidades y desde esta página se puede seleccionar una comunidad para ver todos sus detalles.

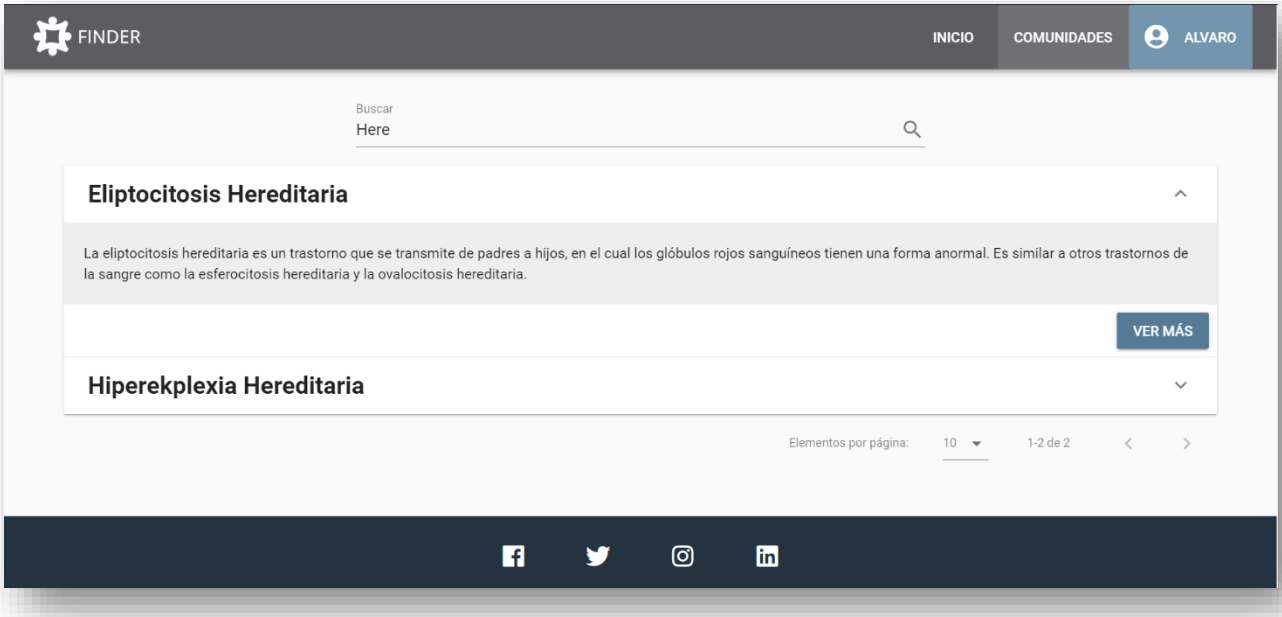

Figura 4.7: Página donde se muestran todas las comunidades

/community: Esta ruta muestra una página con los detalles específicos de una comunidad.

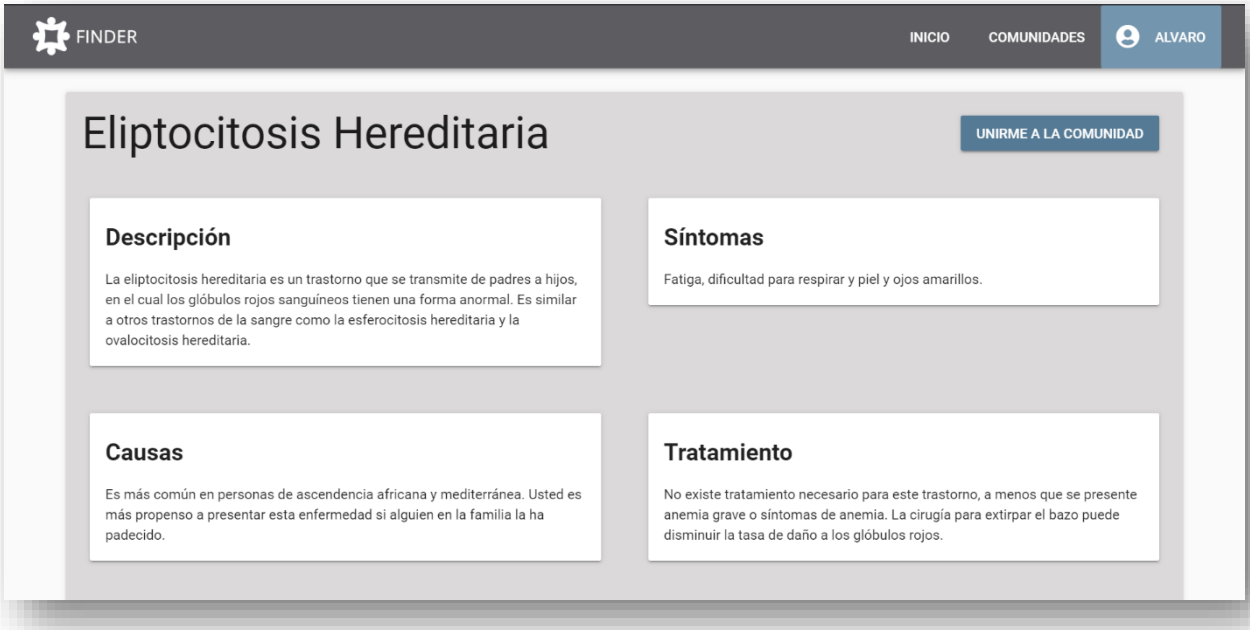

Figura 4.8: Página donde se muestran los detalles de una comunidad específica

/userCommunities: Esta ruta muestra un listado de las comunidades a las está unido un determinado usuario.

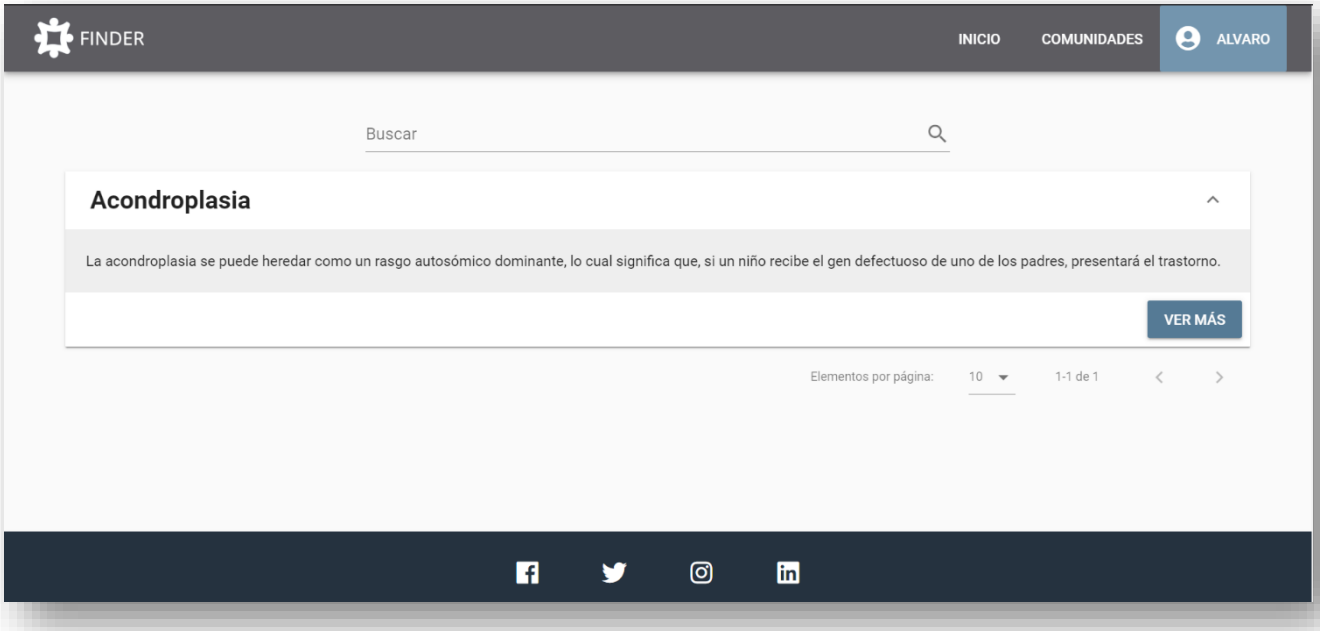

Figura 4.9: Página donde se muestran las comunidades de un usuario

/adminCommunity: Esta ruta muestra un panel de administración de comunidades, en ella se pueden tanto crear, como modificar o eliminar comunidades. Además, también implementa una búsqueda entre comunidades, así como poder ordenar por nombre tanto ascendentemente como descendentemente.

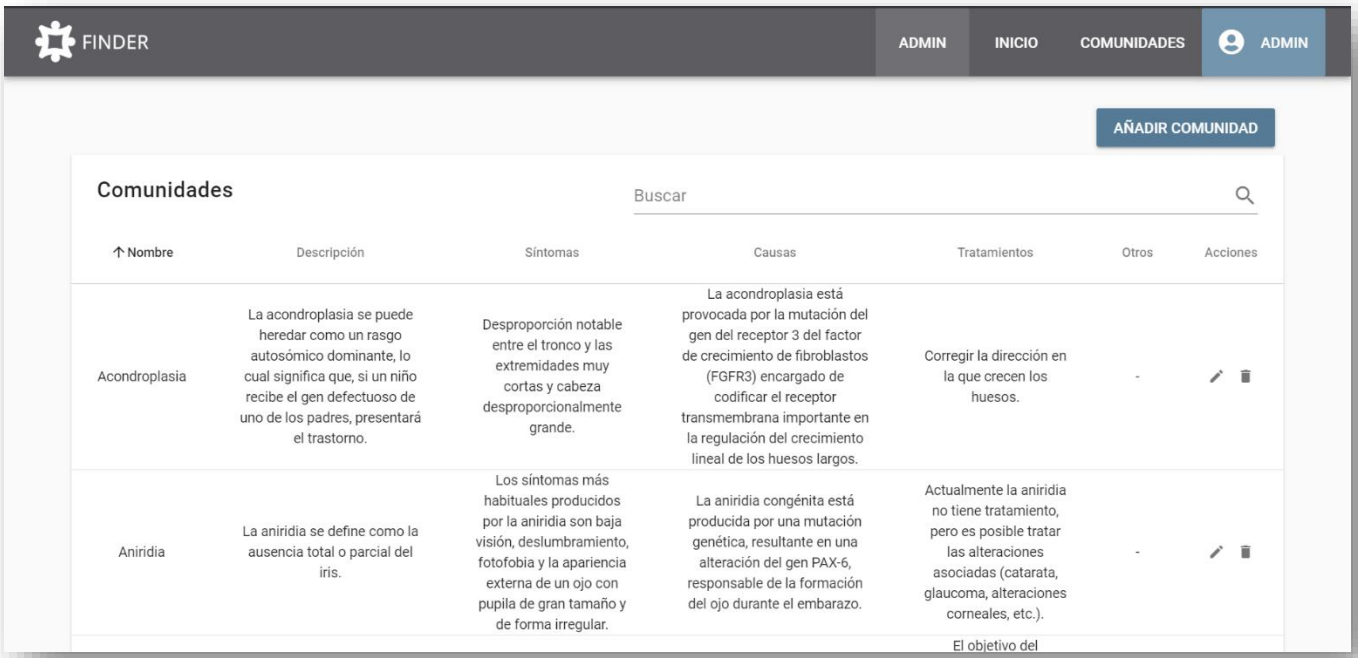

Figura 4.10: Panel de administración de comunidades

Esta ruta está protegida y solo será accesible para las cuentas con rol de administrador. Al iniciar sesión con una de estas cuentas aparecerá una pestaña en la navbar, llamada “Admin” a través de la cual se accederá a la página.

Figura 4.11: Mensaje de error cuando un usuario que no está registrado o no es administrador intenta acceder a él

Al pulsar en crear nueva comunidad o editar una ya existente, aparece un diálogo en pantalla para hacerlo:

Figura 4.12: Ejemplo de editar comunidad en el panel de administración

/adminUser: Similar al panel de administración de comunidades, muestra un listado con todos los usuarios registrados en la plataforma. Desde este panel, se puede dar de alta a un nuevo usuario, modificar uno existente o eliminarlo. De la misma forma, la ruta esta protegida únicamente para usuarios con rol de administrador.

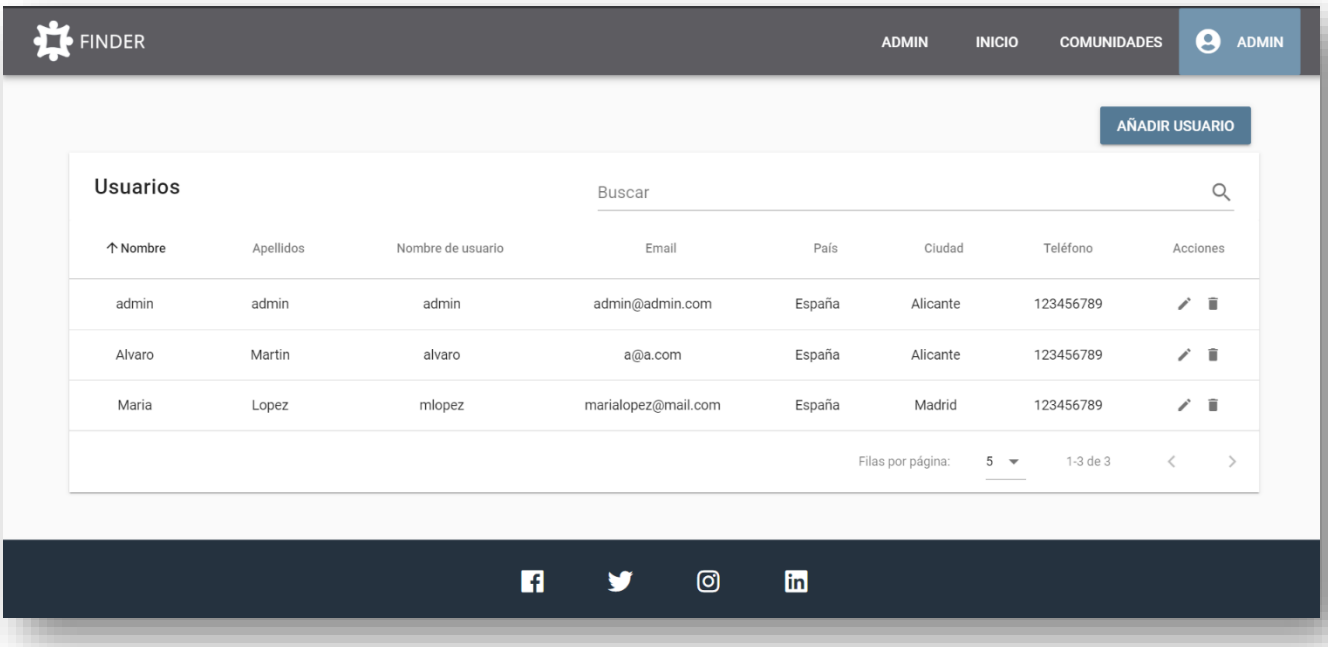

Figura 4.13: Panel de administración de usuarios

Foro

Respecto al foro, en un principio se pensó en utilizar algún componente o herramienta ya existente para implementarlo. Sin embargo, tras varios intentos fallidos debido a que no se encontró ninguno que cuadrara con la arquitectura del proyecto, se optó por crear el foro desde cero.

Se ubica en cada página de cada comunidad, es decir, cada comunidad tiene un foro específico para esa comunidad, donde se podrán crear nuevas discusiones. Estas discusiones se compondrán de un título y una descripción. Una vez creada, dentro de cada discusión se podrán crear nuevas entradas o posts donde responder o simplemente opinar acerca de la discusión.

Para ello, se ha implementado de forma que se muestra en la misma página de una comunidad, en caso de que el usuario esté registrado, si no está registrado se mostrará lo siguiente:

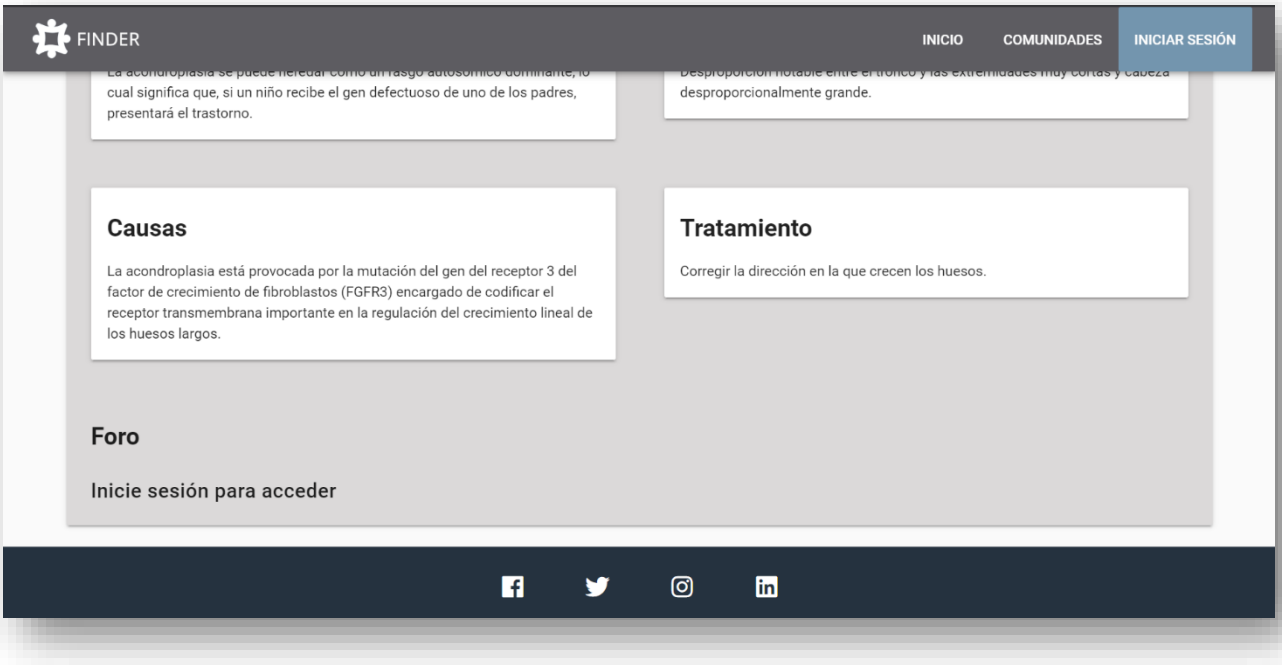

Figura 4.14: Visualización del foro cuando no se ha iniciado sesión

En caso de estar registrado se mostrarán todas las discusiones creadas en esa comunidad en orden descendente siendo las primeras en mostrarse las más recientes en crearse:

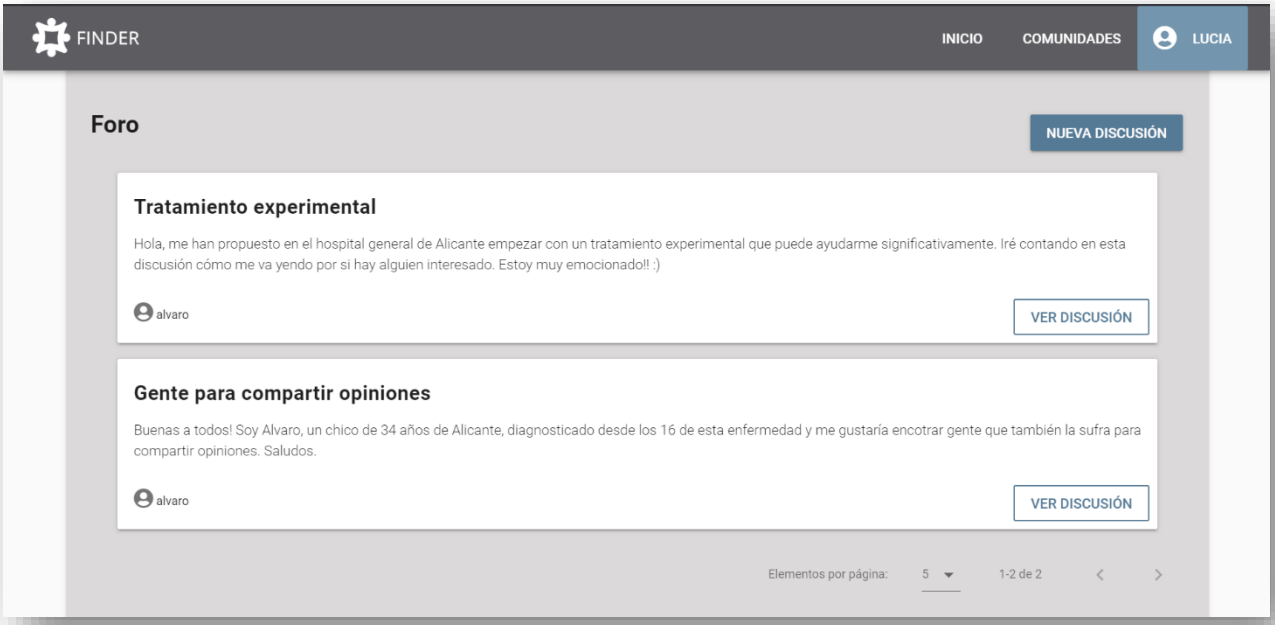

Figura 4.15: Visualización del foro cuando se ha iniciado sesión

Por otra parte, al presionar el botón de “Ver discusión” se abrirá un componente de tipo diálogo donde se mostrarán los detalles de la discusión, así como todos los posts de respuesta que se han hecho dentro de la discusión:

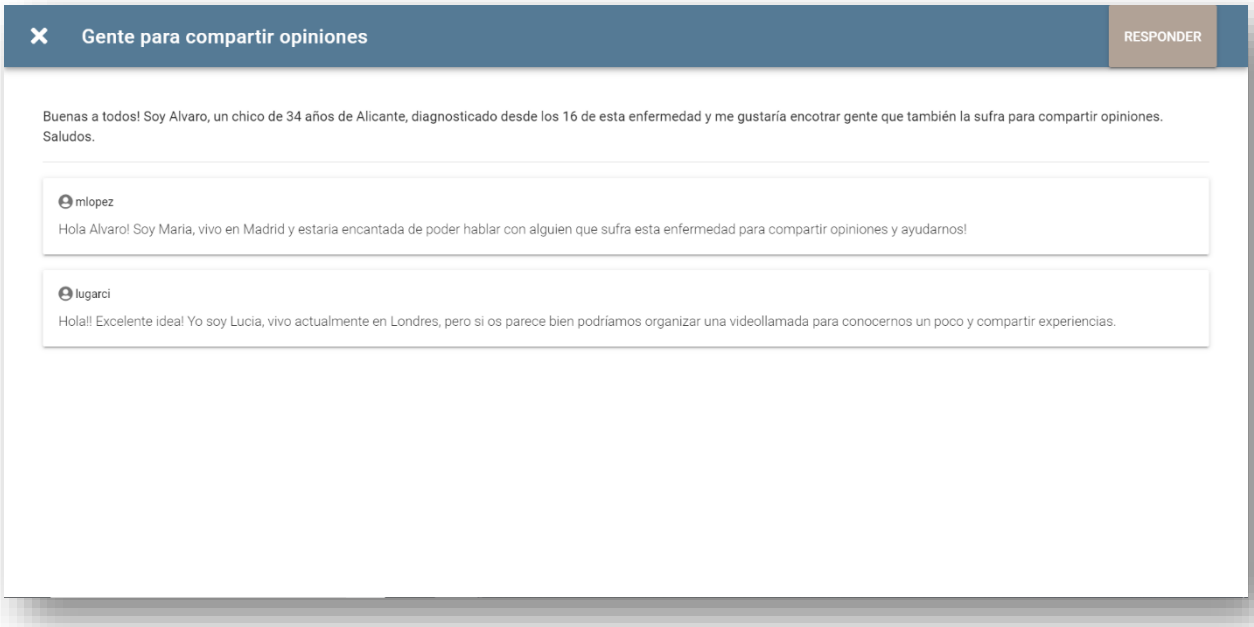

Figura 4.16: Visualización de los detalles de una comunidad

En caso de presionar el botón “Responder” se abrirá otro diálogo para poder añadir un nuevo post a la discusión:

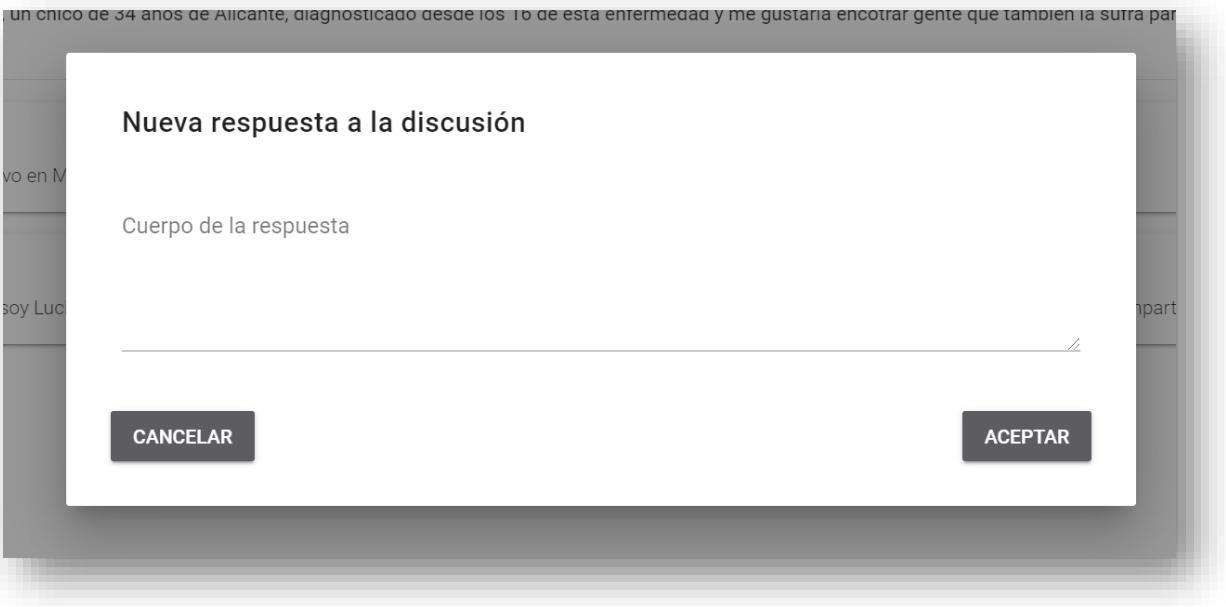

Figura 4.17: Visualización de la opción para crear una nueva entrada en la discusión

De forma similar, si lo que se desea es crear una nueva discusión en la comunidad, el usuario tendrá que presionar el botón de “Nueva discusión”, tras lo que se abrirá un diálogo similar a este último:

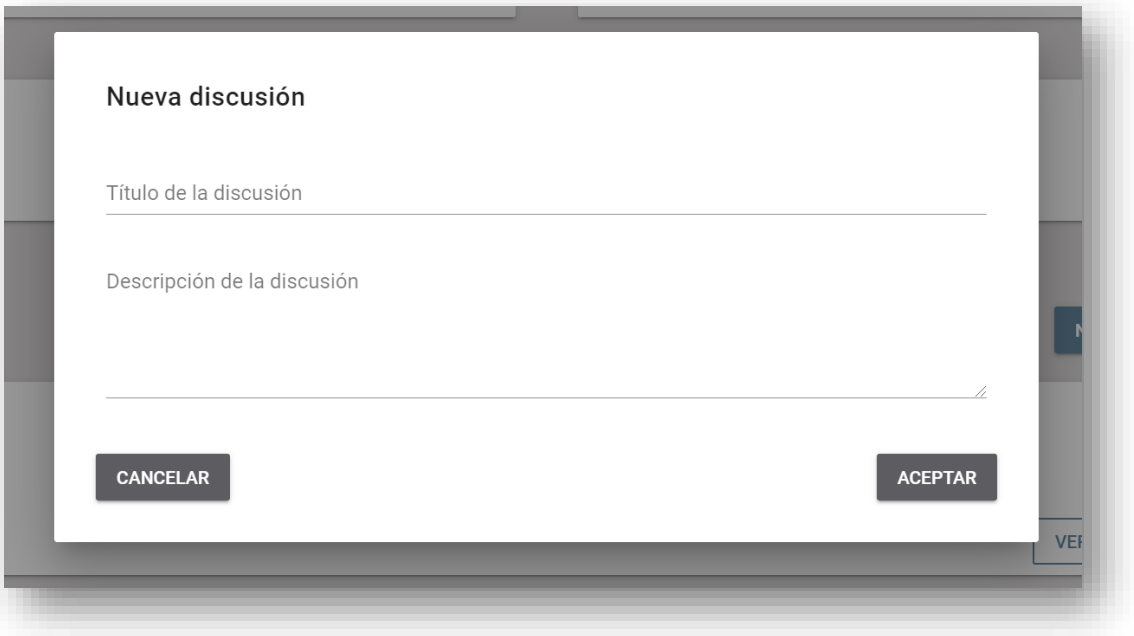

Figura 4.18: Visualización de la opción para crear una nueva discusión

4.2.3 Internacionalización

Se piensa que la internacionalización de la plataforma podría ser un factor fundamental para conectar personas de diferentes países que hablen diferente idioma. Por ello se ha hecho una investigación para averiguar cómo se podría implementar esta funcionalidad con las tecnologías utilizadas.

Para ello, tenemos dos partes bien diferenciadas, internacionalización en el frontend y en el backend. En la parte del frontend, al haberse utilizado Vue.js, encontramos un módulo llamado vue-i18n que es muy útil para implementar esta funcionalidad. Simplemente hay que instalarlo, definir los idiomas en los que estará la página, al igual que los textos que deben traducirse. Esto se ha implementado de la siguiente forma:

Figura 4.19: Código de implementación internacionalización en frontend

Sin embargo, para la parte del backend, implementada en Laravel, sería necesario modificar la base de datos completamente creando tablas auxiliares con las traducciones necesarias para la implementación de la internacionalización, por lo que debido al tiempo limitado del proyecto no ha sido posible implementarlo.

Por lo que ha quedado la estructura preparada en el frontend por si se decide continuar con el desarrollo del proyecto más adelante.

4.2.4 Obtención de datos

Respecto a la obtención de datos fiables para la plataforma, se ha encontrado que el portal Orphanet, que es una fuente de información de referencia sobre enfermedades raras, cuenta con un portal llamado Orphadata[3]. De este portal se puede obtener información sobre todas las enfermedades existentes en la base datos, información que tendría un gran impacto para este proyecto, proporcionando información fiable para los pacientes.

Sin embargo, no ha sido posible obtenerlos debido a que es de pago, costando miles de euros, precio inasumible para este proyecto, pero que quizás si este proyecto se retoma en el futuro, sea asumible.

Orphadata también cuenta con un apartado de información gratuita, pero esta está muy limitada siendo poco más que un índice.

[3] Orphadata: www.orphadata.org

5 Conclusiones

Para poner fin a este proyecto vamos a recapitular acerca de lo que se ha conseguido, mirando los objetivos iniciales, y, por último, añadir una serie de posibles futuras mejoras, que, aunque no se hayan podido implementar debido a que no estaban incluidas en el alcance del proyecto debido a la limitación de tiempo con la que se contaba, serían características que aportarían mucho valor a la plataforma.

Empezando con los objetivos cumplidos, nos encontramos que se ha conseguido crear la plataforma y para ello se ha hecho el estudio de mercado, se han estudiado las tecnologías a utilizar, se ha examinado de dónde se podrían obtener los datos necesarios para la plataforma y se ha diseñado un esquema Entidad-Relación como esquema de base de datos que ha ido variando a lo largo del proyecto. Una vez dados estos pasos, se creó tanto el API con todas las rutas necesarias, así como la parte del frontend, en lo que se ha intentado que sea una interfaz lo más intuitiva y amigable posible.

Respecto a posibles mejoras que sería interesante implementar, las que más impacto se piensa que tendrían y más valor añadirían, serían las siguientes:

- Implementar diferentes categorías para comunidades para que se pueda filtrar por ellas.
- Involucrar a médicos y centros sanitarios, especialistas en diferentes patologías presentes en la plataforma para que puedan ayudar a las personas que lo necesiten.
- Comunicación entre usuarios mediante algún tipo de chat o tecnología similar.

- Monetizar la plataforma para su sustentabilidad, para ello se piensa que quizás tanto centros médicos privados como farmacéuticas podrían pagar por aparecer como recomendados en la plataforma.

- En caso de que la plataforma fuera a salir a producción habría que segurizar las rutas del API para que cada usuario solo tenga acceso a las rutas a las que tenga acceso su rol. Actualmente está segurizado solo para usuario autenticado y usuario no autenticado.

En conclusión, se ha conseguido crear una plataforma robusta. Sin embargo, podría mejorarse ampliamente con las mejoras expuestas anteriormente. Aunque a pesar de que se implementasen éstas seguiría teniéndose que enfrentar a retos muy importantes como conseguir una amplia base de usuarios.

6 Bibliografía

[1] FEDER: https://enfermedades-raras.org/

[2] Orphanet: https://www.orpha.net/

[3] Orphadata: http://www.orphadata.org/

[4] Eurordis: https://www.eurordis.org/

[5] Share4rare: https://www.share4rare.org/

[6] RareConnect: https://www.rareconnect.org/

[7] mydisease2ez: https://www.mydisease2ez.com/

[8] Laravel: https://laravel.com/

[9] Vue.js: https://vuejs.org/

[10] Vuetify: https://vuetifyjs.com/

[11] MySQL: https://www.mysql.com/

[12] Git: https://git-scm.com/

[13] GitHub: https://github.com/

[14] LucidChart: https://www.lucidchart.com/